Comandos usuais para manipulação de string em C++

Aqui estão listados alguns comandos comuns para a manipulação de strings em `C++`. Os exemplos foram feitos com base em perguntas feitas por alunos da sala e podem ser executados no `DevC++`. As soluções aqui apresentadas não são únicas. Verifique o comportamento do código se você mudar as strings que são dadas.

Busca de caractere em string

Objetivo:

Procurar palavras ou letras específicas num texto dado. Diferenciando números, maiúsculas e minúsculas podendo ser um caractere ou um trecho.

Observações:

- A linguagem `C++` entende como caracteres diferentes uma mesma letra quando maiúscula ou minúscula (`a` e `A` são caracteres diferentes, por exemplo). Uma solução para que uma letra maiúscula e minúscula sejam consideradas o mesmo caractere é colocar todos os caracteres maiúsculos ou minúsculos antes da comparação. Isso pode ser feito em um laço que percorre a string e usar as funções `toupper()` (recebe um caractere e retorna seu correspondente maiúsculo) e `tolower()` (recebe um caractere e retorna seu correspondente minúsculo) da biblioteca `cctype`.
- As soluções abaixo utilizam funções prontas disponíveis nas bibliotecas padrão de `C++`. Outra solução possível, seria fazer um laço que percorre uma string comparando com o caractere (ou substring) que estamos procurando.

Comandos possíveis:

- Para `char string`, usando a função `strstr()`

```
#include <stdio.h>
#include <string.h>
#include <iostream>
using namespace std;

int main ()
{
  char string_principal[] ="Uma string de exemplo";
  char substring[] = "string";
  char *pch;
```

```
  // a função strstr() retorna um ponteiro de char que aponta para a posição da primeira
ocorrencia da
  // substring na string_principal
  pch = strstr (string_principal,substring);

  if (pch != NULL){ // caso a string_principal não contenha a substring é retornado um ponteiro
NULL
    cout << substring << " nao esta em " << string_principal << endl;
  }
  else{ // se o ponteiro retornado for diferente de NULL, significa que string_principal contém
substring
    cout << substring << " esta em " << string_principal << endl;
  }

  cout << pch;

  return 0;
}
```

- Para `string`, usando o método `find()`

```
#include <iostream>
#include <string>
using namespace std;

int main () {
    string string_principal = "Uma string de exemplo";
    string substring  = "string";

    size_t pos; // pos é uma variável do tipo size_t
    // size_t é um tipo específico usado para guardar tamanohos e posições na bilioteca string

    // na linha abaixo, o método find busca a substring em string_principal e retorna a posição
    // em que a substring está na string
    pos = string_principal.find(substring);


  if (pos == string::npos){ // caso a string_principal não contenha a substring é retornado o
valor padrão string::npos
    cout << substring << " nao esta em " << string_principal << endl;
  }
  else{ // se pos for diferente de string::npos, significa que string_principal contém substring
    cout << substring << " esta em " << string_principal << endl;
  }

  cout << pos;
```

Mais informações (conteúdos em inglês):

http://www.cplusplus.com/reference/cstring/strstr/

http://www.cplusplus.com/reference/string/string/find/

Concatenar Strings

Objetivo:

Juntar partes de textos para formar um texto maior. Com string tem-se o `strcat(string)`, mas e o problema do espaço em branco que encerra? Pode ser feito algo parecido com array?

Observações:

- Uma forma de concatenar dois arrays `A` e `B` (de qualquer tipo) pode-se seguir o seguinte procedimento:

    1. Criar um array `C` que o tamanho seja a soma do tamanho de `A` e de `B`
    2. Fazer um loop que copia os valores de `A` nas primeiras posições de `C`
    3. Fazer um segundo loop que copia os valores de `B` nas últimas posições de `C`
    4. Ao fim dos dois loops `C` armazenará os valores de `A` e `B` concatenados

Comandos possíveis:

- Para `string`

```
#include <iostream>
#include <string>
using namespace std;

int main () {
    string string1 = "Este e um exemplo de ";
    string string2 = "como concatenar strings";
    string string3;

    // pode-se concatenar variáveis do tipo string utilizando diretamente o operador +
    string3 = string1 + string2;

    cout << string1 << endl;
    cout << string2 << endl;
    cout << string3 << endl;

  return 0;
}
```

Trocar ou excluir caracteres

Objetivo:

Achar só maiúsculas, minúsculas, números, pontuação e, achando-os, como excluí-los ou trocá-los

Observações:

- Para trocar um caractere por outro qualquer, precisamos de saber em que posição ele ocorre (podemos utilizar as estruturas que vimos acima), em seguida atribui-se o novo valor para o vetor nessa posição
- Para excluir um caractere, novamente, precisamos de saber em que posição ele ocorre. Depois, podemos dividir a string em três partes: uma anterior a ser excluída, o trecho a ser excluído, e o trecho posterior ao ser excluído. Por fim, concatenamos o trecho anterior ao posterior, formando uma nova string sem a parte que desejávamos excluir

Comandos possíveis:

- Identificar se um caractere corresponde a uma letra maiúscula: a função `isupper()` da biblioteca `cctype` retorna verdadeiro se o caractere dado em sua entrada for uma letra maiúscula
- Identificar se um caractere corresponde a uma letra minúscula: a função `islower()` da biblioteca `cctype` retorna verdadeiro se o caractere dado em sua entrada for uma letra minúscula

Encontrar tamanho de uma string

Objetivo:

Medir tamanhos de textos dados. Existe um tipo `sizeof(texto)` pra isso?

Comandos possíveis:

- Para `char string`

```
#include <iostream>
using namespace std;

int main () {
    char string[] = "Este e um exemplo de string de tamanho 41";
    int tamanho = 0;

    while (string[tamanho]!='\0'){ // '\0' marca o fim de uma string
        tamanho++;
```

```
    }

    cout << tamanho << endl;

  return 0;
}
```

- Para `string`, usando o método `length()`

```
#include <iostream>
#include <string>
using namespace std;

int main () {
    string string1 = "Este e um exemplo de string de tamanho 41";
    int tamanho = 0;

    tamanho = string1.length(); // O método length retorna o tamanho da string

    cout << tamanho << endl;

  return 0;
}
```

Materiais Complementares

Para aprendermos como utilizar funções já implementadas nas bibliotecas padrão é importante a leitura da documentação da linguagem e fóruns. Infelizmente, a maior parte deste conteúdo está apenas em inglês. Abaixo estão listados alguns conteúdos que podem ser úteis.

Em português:

https://ufsj.edu.br/portal-repositorio/File/prof_ngoulart/notas_aula/AEDS1/string_Cpp.pdf

https://pt.wikibooks.org/wiki/Programar_em_C%2B%2B/Manipulando_strings

Em inglês:

https://www.w3schools.com/cpp/cpp_strings.asp

http://www.cplusplus.com/reference/string/string/